XPose: Reinventing User Interaction with Flying Cameras

Ziquan Lan*, Mohit Shridhar†, David Hsu‡ and Shengdong Zhao§

*‡§NUS Graduate School for Integrative Sciences and Engineering §NUS-HCI Lab

*†‡§Department of Computer Science, National University of Singapore
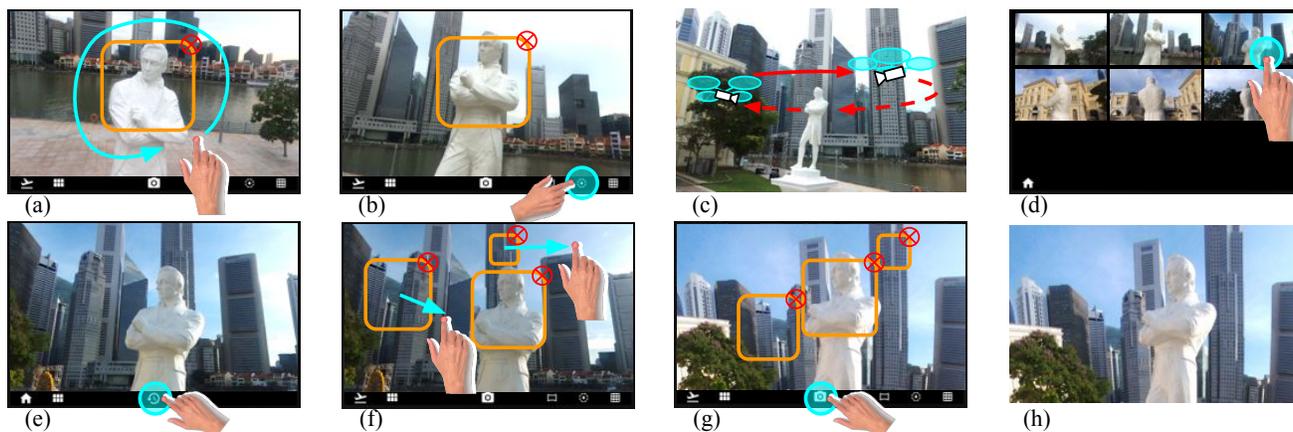
Fig. 1: An envisioned use case. (a) Select an object of interest with the encircle gesture. (b) Activate the *Orbit* exploration mode. (c) The drone-mounted camera takes sample photos while orbiting the object of interest autonomously. (d) Browse sample photos in the gallery preview. (e) Restore a POV associated with a selected sample photo. (f) Compose a final shot by dragging selected objects of interest to desired locations in the photo. (g) Take the final shot. (h) The final photo.

Abstract—XPose is a new touch-based interactive system for photo taking, designed to take advantage of the autonomous flying capability of a drone-mounted camera. It enables the user to interact with photos directly and focus on taking photos instead of piloting the drone. XPose introduces a two-stage *eXplore-and-comPose* approach to photo taking in static scenes. In the first stage, the user explores the "photo space" through predefined interaction modes: *Orbit*, *Pano*, and *Zigzag*. Under each mode, the camera visits many points of view (POVs) and takes exploratory photos through autonomous drone flying. In the second stage, the user restores a selected POV with the help of a gallery preview and uses direct manipulation gestures to refine the POV and compose a final photo. Our prototype implementation, based on a Parrot Bebop quadcopter, relies mainly on a single monocular camera and works reliably in a GPS-denied environment. A systematic user study indicates that XPose results in more successful user performances in photo-taking tasks than the touchscreen joystick interface widely used in commercial drones today.

I. INTRODUCTION

Compared with handheld cameras widely used today, a camera mounted on a flying drone affords the user much greater freedom in finding the *point of view* (POV) for a perfect photo shot. Drone-mounted cameras have produced extraordinary photos, with POVs rarely reachable otherwise [3]. While today drone-mounted cameras remain the toys of hobbyists, in the future many people may carry compact drones in pockets [1] or even wear them on the wrists [4]. They release the drones into the sky for photo taking and use their touchscreen mobile phones as viewfinders. This work develops a prototype flying camera based on a Parrot Bebop quadcopter, with the goal of investigating the underlying user interaction design and system implementation issues.

For interaction design, the envisioned flying camera is conceptually not a drone fitted with a camera, but a camera with flying capability. The difference between the two lies in their mental models of interaction. The former implies a model of interacting with two separate devices, a drone and a camera. The user first pilots the drone painstakingly through a joystick or an emulated joystick interface on a touchscreen (Fig. 2a) in order to reach a desired pose, and then operates the camera to take a photo. Most drone-mounted cameras today adopt this model. In contrast, we seek a unified interaction model for a camera capable of flying. The details of drone control are transparent to the user, making the flying camera more intuitive and easier to use.

To explore the design requirements of flying camera interaction, we started with an interview study with photographers and drone flyers, and identified the main objective of helping the user to explore POVs efficiently while avoiding the perception of latency when the camera transitions between POVs. Our prototype, called XPose, introduces a novel two-stage *eXplore-and-comPose* approach to photo taking in static scenes [1]. In the *explore* stage, the user first selects objects of interest (Fig. 1a) and the interaction mode (*Orbit*, *Pano*, and *Zigzag*) on a touchscreen interface (Fig. 1b). The camera then flies

---

[1]See the XPose video at https://youtu.be/F1hrPb1SlHo.
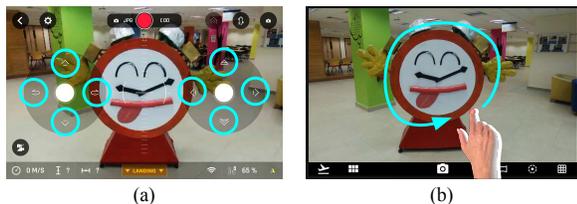
Fig. 2: Compare a common joystick interface and XPose. (a) The left joystick controls the drone's heading direction and altitude. The right joystick controls its translational movement along or orthogonal to the heading direction. (b) With XPose, the user interacts directly with objects in the photo.

autonomously along predefined trajectories and visits many POVs to take exploratory photos (Fig. 1c). The sample photos are presented as a gallery preview (Fig. 1d). The user taps on a potentially interesting preview photo and directs the drone to revisit the associated POV in order to finalize it (Fig. 1e). In the *compose* stage, the user composes the final photo on the touchscreen, using familiar dragging gestures (Fig. 1f). For example, to place a face of a person, according to the rule of thirds [10], the user selects the face and drags it towards the desired location in the photo. To realize this, the camera flies autonomously to a desired POV, instead of relying on the user to pilot manually. XPose also supports manipulation of multiple objects of interest in order to pose them relative to each other in the photo (Fig. 1f). Note that the user may not have a direct line of sight to the flying camera and interacts with it through the touchscreen only.

Our interactive system consists of five main components conceptually: gesture recognition, camera localization, object tracking, trajectory planning, and drone control. A light-weight drone such as the Bebop has severe restriction on the payload and carries only a single forward-facing main camera. One challenge is to localize the camera with respect to the object of interest and track the object reliably, with only a single monocular camera in a GPS-denied environment. We achieve this by leveraging the ORB-SLAM algorithm [28].

This work is a first attempt to address both the interaction design (Sections III and IV) and the system implementation issues (Section V) for a flying camera in a GPS-denied environment. The interplay between interaction design and system implementation is a major challenge for system development. We evaluate XPose in a user study and show that it results more successful user performances in photo-taking tasks than the touchscreen joystick interface widely used in commercial drones today (Section VI). Our current prototype implementation has several limitations. The drone does not yet avoid obstacles autonomously during the flight. Also, as the Bebop has severely limited processing power on-board, most of the computation is performed off-board on a laptop computer. We discuss these issues in Section VII.

II. Related Works

A. Drone Applications

Inexpensive consumer drones have led to many novel applications: running with drones [27], creating flying displays [30,

34], mobile video conferencing [22], and acting as tangible interface building blocks [18]. In particular, systems such as Flying Eyes take videos by tracking a moving subject in sports activities with a drone [20]. Taking such videos, however, differs substantially from taking still photos. For sports or action videos, the main challenge is to track moving subjects in dynamic scenes. For still photos, the key requirements are precise control of POVs and visual composition.

B. Human-Drone Interaction for Drone Navigation

The user may interact with drones through joystick controllers, touchscreens [2, 6], body gestures [11, 31], natural language commands [21], *etc.*. For consumer drones fitted with cameras, the most common interface for drone navigation control is probably a touchscreen with emulated joysticks. See Fig. 2a for an example. The user watches a live video feed from the drone-mounted camera and commands the drone to perform pan, tilt, dolly, truck, and pedestal motions through the joysticks. This approach, which combines joysticks and live video feedback, is also common for teleoperation of remote vehicles [9, 19]. Experiences there suggest that direct manual control through the joysticks faces several difficulties. The low-level motion control afforded by the joysticks are tedious. Further, operating the drone through only the video feed often results in loss of situational awareness, inaccurate attitude judgment, and failure to detect obstacles [26].

An alternative is semi-autonomous supervisory control. Instead of issuing low-level motion commands, the user issues higher-level commands at the task level [35]. This requires the drone to understand the environment, through, *e.g.*, the GPS. GPS-based interfaces enable task-level commands such as tracking [2, 5], orbiting [5, 7], and navigating to waypoints [2, 6]. However, the GPS may be unavailable or unreliable in indoor environments and urban canyons. Further, GPS maps do not supply the visual information required by the user to identify attractive POVs for photo composition. More sophisticated interfaces decouple drone trajectory planning and trajectory execution for photo taking [16, 23]. Trajectory planning occurs offline and assumes a given 3D model of the real-world environment. The decoupled plan-and-execute approach implies a slow interaction cycle and is not quite suitable for photo taking in real time. Acquisition of accurate 3D models is also a major challenge in itself and remains an active area of research [15, 25]. Our work investigates a complete interactive system that aids the user for real-time photo taking. XPose provides a semi-autonomous interface. It leverages a state-of-the art method for 3D mapping and localization in a GPS-denied environment [28], without a 3D model given *a priori*.

III. Interview Study

To explore the requirements of effective user interaction with a drone-mounted camera, we conducted in-depth semi-structured interviews. The interviewees consist of 8 photographers (3 professional and 5 amateur) and 2 professional drone flyers/instructors. For photographers, our questions focused

Intent	Gesture	Camera Motion
Eyeball Panning		Pan and tilt
Translational Panning		Move in the image plane
Zooming		Zoom in and out
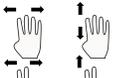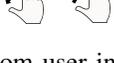
Fig. 3: The mapping from user intents to camera motions.

Orbit Pano Zigzag

Fig. 4: Exploration modes.

on their main considerations when taking photos and how drone-mounted cameras can potentially help. For drone flyers/instructors, our questions focused on their experience with using drones for photo/video taking and how they train novice users. Each interview session lasted from 30 minutes to an hour. All interviews were transcribed into text for qualitative analysis.

As a summary, the photographers' responses suggest a set of well-established considerations for photo taking: POVs, visual composition, lighting conditions, shutter speed, depth of field, *etc.* . They also point to the potential for drone-mounted cameras to discover novel POVs. The drone flyers/trainers' responses include lengthy preparations for drone photo/video taking sessions and extensive training required for novice users. These led to our design objective of a simple, intuitive interface for efficiently exploring many POVs and directly manipulating the visual composition.

IV. EXPLORE AND COMPOSE

With traditional handheld cameras, the user explores POVs by moving around and looking through the viewfinder. Limited by the user's physical movements, the exploration of POVs is *local*, but the visual feedback is almost instantaneous. The joystick touchscreeen interface tries to reproduce this experience for flying cameras: the joysticks control the camera's local movements, and the touchscreen provides visual feedback. However, this approach does not account for the difference in device characteristics between handheld cameras and flying cameras, as well as the resulting system implementation issues. The camera's ability to fly offers the opportunity of exploring POVs more *globally*. At the same time, it is more difficult for a flying camera to achieve an intended POV precisely, due to air disturbance and other factors. Further, visual feedback is not instantaneous, because of the communication delay between the flying camera and the user's mobile device.

We introduce the two-stage explore-and-compose approach, which enables the user to explore a wide range of POVs efficiently in a hierarchical manner. In the explore stage, the user samples many POVs globally at a coarse level, through autonomous drone flying. In the compose stage, the user chooses a sampled POV for further refinement and composes the final photo on the touchscreen by interacting directly with objects of interest in the image.

We now illustrate our approach in detail with a concrete scenario (Fig. 1). Our main character, Terry, walks along a river on a sunny day and stops at a white statue. She wishes to take a photo of the statue, but can hardly get a close-up shot with a handheld camera, as the statue is almost 15 feet in height. Terry launches XPose using the associated app on her mobile device. The home view of the app contains a viewfinder, which displays the live video feed from the drone-mounted camera.
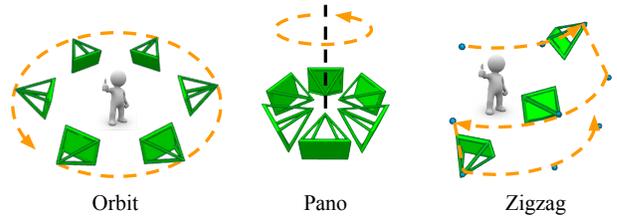
A. Explore

Terry is initially unsure about the best viewing angle for the shot and decides to explore the POVs around the statue. She selects the statue as the object of interest and uses XPose's exploration modes to sample the POVs.

1) Object of Interest Selection: First, Terry performs pan and zoom gestures (Fig. 3) on the touchscreen to get the statue into the viewfinder. Then she draws a cirle around the statue in the viewfinder (Fig. 1a). A rectangular bounding box appears to confirm that the statue has been selected and is being tracked.

2) POV Sampling: While the statue is being tracked in the viewfinder, Terry activates the *Orbit* exploration mode (Fig. 1b). The flying camera then takes sample shots while orbiting around the statue autonomously (Fig. 1c).

XPose currently provides three explorations modes: *Orbit*, *Pano*, and *Zigzag* (Fig. 4). They leverage the autonomous flying capability of a drone-mounted camera and systematically explore the POVs by taking sample shots evenly distributed along predefined trajectories. The Orbit mode is useful when the user has a single main object of interest, *e.g.*, a person, a statue, or an interesting artifact, but is quite unsure about the viewing angle for the best shot. Under this mode, the camera flies a full circle, while looking inward at the object in the center. Under the Pano mode, the camera looks outward, instead of inward. The camera stays at a fixed location, while panning horizontally for full 360 degrees. This mode is well-suited for panoramic shots of scenic landscapes, such as oceans, prairies, or glaciers. The Orbit and Pano modes pan the camera, but fix its tilt. The Zigzag mode exploits both panning and tilting. It is useful when the user knows roughly the best viewing angle. The cameras flies along circular arcs at multiple heights, all centered at an object of interest. Again, the camera always points at the object. One use of the Zigzag mode is to take a selfie against a scenic background. The exploration modes free the user from the tedium of manually piloting the drone through low-level motion commands.

B. Compose

After getting many sample shots through the Orbit mode, Terry is ready to finalize the photo.

System Components	Explore		Compose	
	Object of Interest Selection	*POV Sampling*	*POV Restore*	*Direct View Manipulation*
Gesture Recognition	✔			✔
Camera Localization		✔		✔
Object Tracking	✔	✔*		✔*
Trajectory Planning		✔	✔	✔
Drone Control	✔	✔	✔	✔
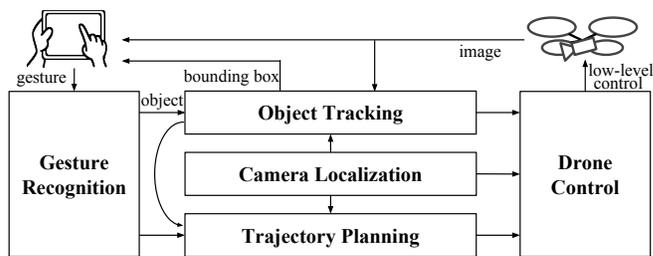
Fig. 5: XPose. (a) Main functions and system components. ✔ indicates a required system component. ✔* indicates that the system component is required only if there are user selected objects of interest. (b) An overview of the system architecture.

1) POV Restore: Terry switches to the gallery view in the app and browses the sample shots displayed there (Fig. 1d). All sample photos have the statue in the center, but different backgrounds. To take a closer look, Terry taps on a photo to see it in the full-screen photo view (Fig. 1d,e). One photo with many tall buildings in the background looks promising, but the composition is not ideal. The accidental alignment of the statue's head and a tall building is distracting. To refine it, Terry taps a button and commands the flying camera to restore the POV associated with the selected sample photo (Fig. 1e).

2) Direct View Manipulation: From the restored POV, Terry selects two buildings in the viewfinder as additional objects of interest and drags them, one at a time, to the left and right side of the statue respectively, so that the statue's head appears in the gap between the two buildings (Fig. 1f). XPose flies to a new POV that produces the desired composition as closely as possible and displays the photos in the viewfinder. Quite satisfied, Terry takes the final shot (Fig. 1g).

V. SYSTEM IMPLEMENTATION

XPose system consists of five tightly integrated main components: gesture recognition, camera localization, object tracking, trajectory planning, and drone control. Fig. 5 shows the main system functions and components, as well as an overview of the system architecture. In this section, we first give an overview and then provides some details on each component.

A. Overview

The table in Fig. 5a shows how the system functions depend on the components. Essential components such as drone control are required for all functions.

1) Object of Interest Selection: For object selection, the system must recognize the user's gestures: pan, zoom, encircle, *etc.*. For the pan and zoom gestures (Fig. 3), the system executes the corresponding motions. For the encircle gesture (Fig. 1a), the system selects the object and tracks it in the image, as the flying camera moves. An image-based tracking algorithm [24, 29] seems a natural choice for object tracking. However, existing image-based tracking algorithms are not robust against large viewing angle changes (*e.g.*, in the Orbit mode) or large viewing distance changes (*e.g.*, while zooming in and out). We present our approach to robust object tracking in Section V-D.

2) POV Sampling: For POV sampling, the system first plans an exploratory trajectory according to the selected exploration mode and samples the POVs at equal distance along the trajectory. It visits each sampled POV sequentially, takes a photo at the POV, and stores both location and image. The system must be localized at all times, in order to check whether a planned POV has been reached.

3) POV Restore: The system keeps track of the POVs of all sample photos obtained in the explore stage. When the user asks to restore the POV of a selected sample photo, the system plans a restoring trajectory and executes it until it reaches the designated POV.

4) Direct View Manipulation: To finalize the composition, the user may select objects of interest and drag them to desired locations in the photo. The system must recognize these gestures. It then computes the POV that produces the desired composition as closely as possible, plans a trajectory to it, and flies there.

B. Gesture Recognition

XPose uses Android's standard gesture detection library to detect three main types of gestures on the touchscreen: pan and zoom gesture for finding objects of interest (Fig. 3), encircle gesture for object selection (Fig. 1a), and drag-and-drop gesture for direct view manipulation (Fig. 1f).

C. Camera Localization

Unlike handheld cameras, our flying camera explores POVs globally. It must have greater awareness of the surrounding environment, and localization becomes a crucial issue. XPose uses a state-of-the-art monocular visual SLAM algorithm, ORB-SLAM [28], to build a SLAM map and localize the camera with respect to a fixed world coordinate. Localization provides crucial support for trajectory execution and object tracking.

D. Object Tracking

For robust object selection and tracking, we exploit ORB-SLAM and use the sparse points produced by the algorithm to represent objects. ORB-SLAM is a feature-based visual SLAM system. It provides 2D-to-3D point correspondence: each 2D feature point exacted from the image is associated with a 3D map point in the SLAM map.

For object selection, the 2D feature points encircled by the user's stroke and their corresponding 3D map points are used to represent an object.

For object tracking, we need to display a bounding box around each selected object on the image as a visual cue. For simplification, we first compute the centroid of each object by taking a weighted average of its 3D map points. The weight of a map point is inversely proportional to the distance between its corresponding 2D feature point and the center of the selection stroke's bounding box. The idea is such that the points closer to the center of the selected region are more important. Then, the 2D projection of the centroid on the image plane is used as the center of the bounding box. The size of the bounding box is continuously estimated by computing the distance from the camera to the object centroid.

The center of the bounding box is a simplified representation of an object's composition location. Formally, the composition of an object ω is the region it occupies on the image, denoted by $\mathbf{R}_\omega$, and its desired composition is another region $\mathbf{R}_\omega^*$ on the image. We define *composition error* ϵ_c as the difference between the actual composition and the desired composition:

$$\epsilon_c = \sum_{\omega \in \Omega} d_H(\mathbf{R}_\omega, \mathbf{R}_\omega^*) \tag{1}$$

where Ω is a set of objects and $d_H(\cdot, \cdot)$ is the Hausdorff distance between two sets. The Hausdorff distance is normalized with respect to the diagonal length of the image. For simplification, we use the center of the bounding box as the actual composition location, and the ending point of the direct manipulation gesture as the desired composition location.

This object-tracking implementation, based on ORB-SLAM, is robust against many practical issues common to drone-mounted cameras, such as temporary frame drops, occlusions, camera exposure changes, *etc.* . Unlike (semi-) dense SLAM algorithms (*e.g.*, [12, 13, 32]), ORB-SLAM tracks a relatively sparse set of features, but it is sufficient for object tracking in our experience, provided, of course, enough feature points are extracted from the objects of interest.

E. Trajectory Planning

The system generates several types of trajectories for POV sampling, POV restore, and direct view manipulation, respectively.

1) POV Sampling: We now describe POV sampling trajectory under each exploration mode.

Orbit. To start the Orbit mode, XPose requires a main object of interest to be selected. The Orbit mode generates a full circular trajectory of POVs looking inward at the object. Formally speaking, the circle is a essentially line of latitude on a sphere, of which the center is the object centroid in the map, and the radius of the sphere is the distance from the object centroid to the camera position where the Orbit mode starts. For each camera position along the circle, a camera orientation is computed to maintain the object composition on the image plane.

The sample shots are planned to be taken evenly along the circle. However, since a drone can hardly reach a planned POV exactly, the sampling condition is relaxed from reaching an exact POV to entering a region of satisfied POVs. The region is bounded by various factors: the composition error, the distance difference to the object centroid, the latitude and longitude angle differences of the sphere centered at the object.

Pano. The Pano mode does not requires any object of interest to be selected. It generates a trajectory of POVs by spinning around at a fixed point. In other words, the POVs have the same camera position and tilt angle, but different camera pan angles.

The sample shots are taken at evenly distributed pan angles. Again, a region of satisfied POVs is used, which is bounded by the position displacement and the orientation difference.

Zigzag. The trajectory generation and execution of the Zigzag mode are very similar to that of the Orbit mode. The Zigzag mode also requires a main object of interest to be selected. The major difference is the sampling pattern. Considering the sphere centered at the object, the Zigzag mode samples POVs on a patch of the sphere by deviating locally from the camera position where the Zigzag mode starts. The patch is discretized as multiple circular arcs along different latitudes of the sphere. During execution, the drone moves along each arc one by one.

2) POV Restore: For POV restore, the trajectory is simply a line segment connecting the current POV and the target POV to be restored, without considering potential collisions. More sophisticated collision-free trajectory generation could be used, but is beyond our current focus in this work.

The end of a POV restoring trajectory is a region of POVs around the target POV, which is bounded by the position displacement and the orientation difference.

3) Direct View Manipulation: For direct view manipulation, the trajectory is also a line segment, but connecting from the current POV to a POV that minimizes the composition error (Equation (1)). This is essentially an optimization problem under geometric constraints: the trajectory planner needs to find a POV such that the objects of interest are located at the user desired positions on the image. Existing methods [17] may provide POVs that are unreachable, *e.g.*, very far away or colliding with obstacles. We propose a sample-based method to solve the problem with an approximate POV, which is easy to implement. The idea is to sample multiple candidate POVs and pick the best one. However, the space of POVs is very large. To achieve real time performance, we reduce the sampling space by decoupling the camera position and the camera orientation. We only sample camera positions, and at each sampled camera position we find an approximately optimal orientation. The details are described below.

First, the method samples many camera positions as candidates around the current POV. Those camera positions are sampled unevenly: it is denser near the current POV and sparser further away (with an upper bound). This sampling strategy is to increase the chance of finding good nearby POVs in order to minimize the potential flying distance. Second, the

candidate orientation at each candidate position is estimated by averaging multiple orientations. Each is the optimal orientation at that candidate position for one object. At last, the candidate POV with the minimum composition error is returned.

The POV found by the proposed method is an approximate solution that may not be globally optimal. However, since the dragging gestures are imprecise touch inputs after all, it is unnecessary to over-interpret those gestures very accurately. More importantly, this method can be directly integrated with other sample-based planning methods that generate collision-free trajectories in more complicated environments.

F. Drone Control

XPose is implemented based on a Parrot Bebop Drone, with a forward-facing on-board camera. The camera has a built-in digital stabilization module to ensure the camera's up vector is always orthogonal to the ground and pointing upwards. The drone is controlled with 6 Degrees-of-Freedom (DoFs) control commands. The control commands set the reference values for the on-board firmware to control the following 6 DoFs: the roll angle, the pitch angle, the yaw rotational velocity, the vertical velocity and the camera's pan and tilt angles.

We use 6 independent PID controllers for each DoF. The control gains are tuned by assuming that the SLAM map is in metric unit. To resolve the issue of scale ambiguity, we adopted a maximum likelihood scale estimation method using the on-board ultrasonic altitude measurement [14].

XPose controls the drone to execute the planned trajectories. Each trajectory consists of a series of POVs in 5 DoFs, with 3 DoFs in camera position and 2 DoFs in orientation (pan and tilt). Given a reference POV from the trajectory and the current POV from the camera localization, the 3-DoF reference camera position is achieved by controlling the roll and pitch angles, and the vertical velocity. The reference tilt is achieved by controlling the camera's tilt. The reference pan is jointly controlled by the yaw rotational velocity and the camera's pan.

During trajectory execution, the POV estimated by ORB-SLAM cannot be directly used to control the drone, due to the delay caused by video transmission ($\sim$200ms) and processing ($\sim$20ms). A delayed estimation of POV may cause oscillating behaviors while controlling the system at a high update rate. To compensate the delay, a Kalman Filter is used to predict the true POV with a constant-velocity model. This predicted POV is then used for drone control during trajectory execution. While our method serves the purpose of system evaluation (Section VI), there are more sophisticated methods [14] if needed.

VI. EVALUATION

We conducted a user study to compare XPose and a joystick interface, in order to examine the feasibility of XPose and quantify the performance differences between the two.

A. Experimental Setup

The study consists of two sets of experiments. The first set focuses on interaction design. It separately evaluates XPose's design of POV exploration and of visual composition. It tries to minimize the effects of system implementation by placing the flying camera in a motion capture studio, which provides highly accurate camera localization and object tracking. The second set of experiments focus on the overall system performance in a more realistic setting, without the help of a motion capture system.

We compare XPose with one of the Parrot Bebop's native interfaces, the *Joypad* mode in the *FreeFlight Pro* app. Joypad emulates a joystick interface on a touchscreen and provides the user low-level manual control of the drone (Fig. 2a). We use it as the baseline for comparison, as joysticks and emulated joystick interfaces are the most widely used method in commercial drones available today.

B. System Hardware and Software Setup

1) Hardware: Our experiments use a Parrot Bebop drone with a built-in forward-facing camera. The user's mobile device is an ASUS Google Nexus 7 (2012) tablet with a 7.0" multi-touch display, running Android version 5.1. Both the drone and the tablet are connected via Wi-Fi to an ASUS laptop PC with an Intel Core i7 processor and 16 GB RAM. For the first set of experiments, the PC is also connected via Ethernet to a server hosting the motion capture system. Our VICON motion capture system consists of 10 cameras mounted on the ceiling and provides accurate location information of the drone and other tracked objects at 50 Hz.

2) Software: We use an open-source drone driver, Bebop Autonomy [8], to communicate with the Bebop at 30 Hz. The Bebop runs low-level controllers on-board. The tablet runs Rosjava and detects gestures using Android's gesture detection library. The PC handles all other computations. It also hosts the Robot Operating System framework, which is used to establish communication among the drone, the tablet, and the laptop PC.

C. Interaction Design Evaluation

1) Evaluation of POV Exploration: The explore stage aims to find potentially interesting POVs. We designed a task to evaluate the effectiveness of POV exploration in a controlled lab setting. The setup consists of a transparent board with several target regions of interest on both sides (Fig. 6a). Each target region is made of a piece of circular cardboard and contains several text labels pinned at different orientations. One can see the text on the label only from a certain viewing angle range. The text on each label is unique. In each trial, the flying camera is initially placed at a fixed location on the floor 2 meters away from the board. The participant is instructed to face away from the flying camera. S/he interacts with the flying camera through the tablet only and has no direct line sight. According to our interview study, drones often fly out of the direct sight of operators. The participant is asked to read out all the text labels on the board as fast as possible. The trial terminates when the participant believes that s/he has finished reading all the text labels and lands the flying camera. The trial pauses, if the flying camera crashes. The drone is placed on the
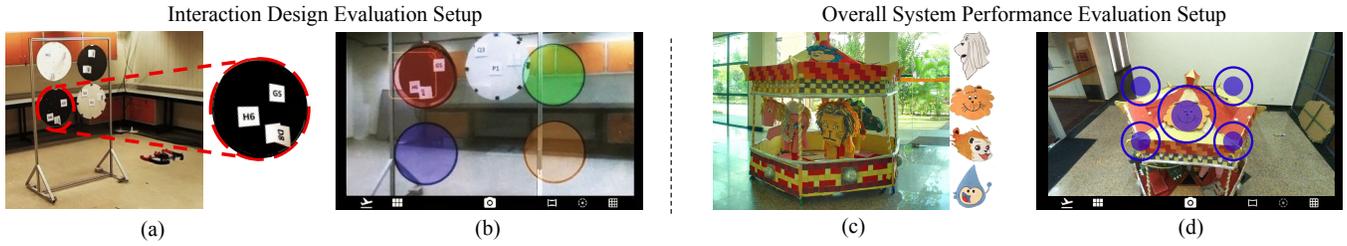
Fig. 6: Evaluation scenes. (a) Evaluation of POV exploration. Text labels are pinned at different orientations on the board. (b) Evaluation of photo composition. Four translucent colored circles indicate possible composition locations in the photo. (c) Evaluation of overall system performance for photo taking. The enlarged cartoon figures are shown next to the merry-go-around in a semi-outdoor environment. (d) Each double circle indicates a possible composition location for the cartoon figure, and its size indicates the desired size of the figure in the photo.

floor at the crash site. The trial resumes after the participant relaunches the drone and continues with the task.

We measured the *completion time* for each trial and the *coverage*, *i.e.*, the percentage of correctly reported text labels.

2) Evaluation of Visual Composition: Once a potential POV is identified, the compose stage aims to place objects of interest at desired locations in the photo by refining the POV. We designed another evaluation task for composition, with a similar setup. In each trial, the participant sees in the viewfinder four circles of different colors, placed in the photo frame according to the rule of thirds (Fig. 6b). S/he is asked to put a specified target region at the location of a randomly assigned circle so that they match as well as possible.

Again, we measured the *completion time* for each trial as well as the *composition error* (Eq. (1)), which provides one measure of composition quality.

3) Participants: 8 volunteers (2 female, 6 male, aged 23–30) were recruited from the university community and the IT industry. All participants had prior experience taking photos, and 3 had experience flying drones.

4) Within-Participants Design: Each participant used each interface to perform each of the two evaluation tasks three times. The order of trying the two interfaces was counterbalanced. The two tasks were ordered sequentially, as we were not interested in comparing between different tasks. Before the experiment began, each participant was instructed for 10 minutes for each interface to get familiar with it.

5) Results: We conducted two-way repeated measure ANOVA to analyze the results.

POV Exploration. The difference between XPose (93.1%) and the joystick interface (93.6%) in POV coverage was not significant (all $p > 0.05$). See Fig. 7a. The participants were able to identify most text labels (POVs) using either interfaces.

However, Fig. 7b shows that XPose (85.4s) was significantly faster ($F_{1,7} = 46.36$, $p < 0.001$, $\eta^2 = 0.87$) than the joystick interface (153.9s). We also observed a significant trend ($F_{2,14} = 14.72$, $p < 0.001$, $\eta^2 = 0.68$) on completion time over the trials, indicating a learning effect for participants for both interfaces. Since most participants were not familiar with drone flying, this effect was expected. However, there was no significant interaction between the effect of interface and that of trial on the completion time ($p > 0.05$), suggesting that the learning curves of the two interfaces are similar.

Visual Composition. The composition error using XPose (0.86%) was smaller than that of the joystick interface(1.28%). See Fig. 7c. While the difference was statistically significant ($F_{1,7} = 21.336$, $p = 0.002$, $\eta^2 = 0.75$), both errors were about 1% and unlikely to make much difference in most photos. We did not observe any other main effects or interaction effects (all $p > 0.05$).

Again, Fig. 7d shows that XPose (34.2s) was significantly faster ($F_{1,7} = 24.36$, $p = 0.002$, $\eta^2 = 0.78$) than the joystick interface (47.8s). The participants also became significantly faster over trials ($F_{2,14} = 7.65$, $p = 0.006$, $\eta^2 = 0.52$), and there was no significant interaction between the effect of interface and that of trial on the completion time ($p > 0.05$).

Overall, XPose was significantly faster than the joystick interface in both POV exploration and photo composition, while achieving a comparable level of task performance measured in POV coverage or composition error. Although the number of participants in the study is relatively small, the confidence intervals are clearly separated (Fig. 7b,d).

D. Overall System Performance Evaluation

We conducted the second set of experiments to evaluate the overall system performance in a more realistic photo-taking setting, by removing the motion capture system.

1) Evaluation of Photo Taking: We set up a merry-go-round in a semi-outdoor environment and hid cartoon figures inside or on top of the merry-go-round (Fig. 6c). The objective is to find a specified cartoon figure and compose the photo suitably. In each trial, the flying camera is initially placed at a fixed location on the floor 3 meters away from the merry-go-around. As usual, the participant has no direct line of sight of the flying camera or the merry-go-round. The participant is asked to take a photo of a specified cartoon figure and compose the photo so that the figure appears at one of the five locations marked by double circles in the viewfinder (Fig. 6d). Further, the figure must fully cover the inner circle and be fully enclosed by the outer circle. The trial starts after the participant launches the drone and terminates when the participant takes a shot. Each trial uses a different cartoon figure.

We measured the task *completion time* for each trial and the *success rate*.
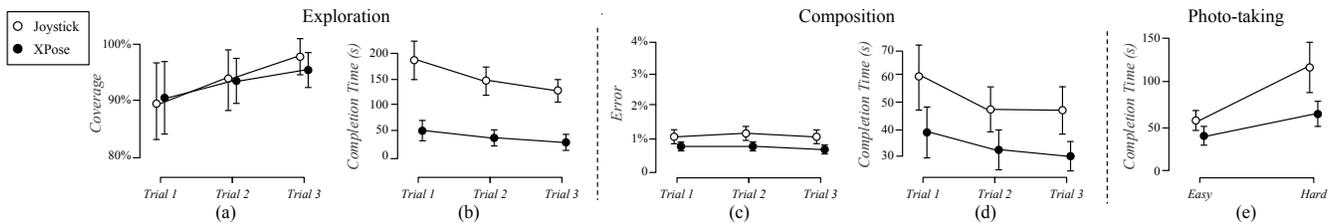
Fig. 7: Performance comparison between the joystick interface and XPose. Error bars indicate 95% confidence intervals.

2) Participants: 8 new volunteers (2 female, 6 male, aged 22-30) were recruited from the university community. None participated in the earlier experiments. All participants had experience taking photos, and 4 had experience flying drones.

3) Within-Participants Design: Each participant used each interface to perform the task with two difficulty levels, one having an easy-to-find figure and one having a harder-to-find figure. The order of trying the interfaces was counterbalanced.

4) Results: The success rate was 100%, as all specified cartoon figures were found and composed as required.

We conducted two-way repeated measure ANOVA to analyze the completion time (Fig. 7e). XPose (57.1s) was significantly faster ($F_{1,7} = 19.12$, $p = 0.003$, $\eta^2 = 0.73$) than the joystick interface (85.3s). As expected, completing the easy task was significantly faster ($F_{1,7} = 52.79$, $p < 0.001$, $\eta^2 = 0.88$) than completing the hard task. Interestingly, there was significant interaction ($F_{1,7} = 6.10$, $p = 0.043$, $\eta^2 = 0.47$) between the effect of interface and that of difficulty on the completion time. Increased difficulty caused a large increase in the completion time for the joystick interface, but it caused a smaller increase for XPose. We propose the following reason. In the more difficult task, the cartoon figure was partially occluded. XPose provided a gallery preview so that participants could examine each sample shot closely and find the partially occluded figure faster.

VII. Discussion

The evaluation results show that XPose outperforms the joystick interface in POV exploration, in visual composition, and in photo taking. The joystick interface forces the user to pilot the drone manually through low-level motion commands. Doing so while searching for a good POV at the same time is difficult and tedious. Further, the communication delay between the flying camera and the touchscreen mobile device often causes the camera to overshoot the desired pose. Our explore-and-compose approach demonstrates great potentials for photo-taking with flying cameras. While our experiments tested only a small range of representative tasks, we are pleased to see that our interaction design enabled more efficient POV exploration with predefined exploration modes and easier photo composition with direct manipulation of objects of interest. POV exploration and photo composition are essential sub-tasks for photo taking. Creating more efficient and user-friendly approaches to these tasks contribute significantly to better photo taking. Participants expressed that XPose is more natural and intuitive to use: *"Yours (XPose) is easier to use as I can focus on the task instead of flying the drone."*

More encouragingly, the two stages, explore and compose, worked well together for a more realistic photo taking task in a GPS-denied semi-outdoor environment. Together, the results of our user study suggest that XPose clearly represents a step forward towards a practically useful system.

From the interaction design perspective, the explore-and-compose approach currently focuses on photo taking in static scenes. We believe that the underlying design idea is useful for dynamic scenes as well, but the interaction design and the system implementation become more challenging. In addition, the explore-and-compose approach could also be adopted in other modalities, such as mouse-based interfaces.

From the system implementation perspective, our current prototype works successfully in indoor, semi-outdoor, and limited outdoor settings, with two main limitations. First, it assumes an environment relatively free of major obstruction for drone flying. By combining the SLAM map and the collision avoidance capability [33], we can relax this assumption. Additional sensors, such as the ultrasonic range finder, would also help. Second, we rely on a laptop computer for most of the required computation, because the Parrot Bebop has very limited on-board processing power. With rapid advances in hardware technology and decreasing cost, we expect to overcome this limitation in the near future.

VIII. Conclusion

XPose introduces the explore-and-compose approach to photo taking and provides a unified interaction model for flying cameras. It enables a low-cost quadcopter to fly in a GPS-denied environment and provides intuitive and effective interactions that aid the user for photo taking in real time. As a result, the user focuses on taking photos, instead of piloting the drone. Our experience highlights the importance of integrating interaction design and system implementation. Good interaction design must be rooted in realistic assumptions of system capabilities, and new system implementation technologies open up opportunities for innovative interaction design. While our prototype implementation still has several limitations, it is a first step towards the ultimate vision of a flying camera for everyday use.

REFERENCES

[1] AirSelfie. URL http://www.airselfiecamera.com/.

[2] DJI drones. URL http://www.dji.com/products/drones.

[3] Dronestagram. URL http://www.dronestagr.am/2016-international-drone-photography-contest.

[4] Nixie. URL http://flynixie.com.

[5] Hexo+. URL https://hexoplus.com/.

[6] Parrot bebop drone. URL http://global.parrot.com/au/products/bebop-drone.

[7] 3DR Solo. URL https://3dr.com/solo-drone.

[8] AutonomyLab. Autonomylab/bebop_autonomy. URL https://github.com/autonomylab/bebop_autonomy.

[9] P. Ballou. Improving pilot dexterity with a telepresent rov. In *Proc. the Vehicle Teleoperation Interfaces Workshop, IEEE ICRA*, 2001.

[10] S. Caplin. *Art & design in Photoshop*. Elsevier/Focal, 2008.

[11] J. R. Cauchard, K. Y. Zhai, J. A. Landay, et al. Drone & me: an exploration into natural human-drone interaction. In *Proc. ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*, pages 361–365, 2015.

[12] A. Concha and J. Civera. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 5686–5693, 2015.

[13] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular slam. In *Proc. European Conf. on Computer Vision*, pages 834–849, 2014.

[14] J. Engel, J. Sturm, and D. Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Proc. Robotics and Autonomous Systems*, 62(11):1646–1656, 2014.

[15] J. Fuentes-Pacheco, J. R. Ascencio, and J. M. Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43:55–81, 2012.

[16] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges. Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals. In *Proc. CHI'16 Conf. on Human Factors in Computing Systems*, pages 2508–2519, 2016.

[17] M. Gleicher and A. Witkin. Through-the-lens camera control. In *Proc. ACM SIGGRAPH Computer Graphics*, volume 26, pages 331–340, 1992.

[18] A. Gomes, C. Rubens, S. Braley, and R. Vertegaal. Bitdrones: Towards using 3d nanocopter displays as interactive self-levitating programmable matter. In *Proc. CHI'16 Conf. on Human Factors in Computing Systems*, pages 770–780, 2016.

[19] D. W. Hainsworth. Teleoperation user interfaces for mining robotics. *Autonomous Robots*, 11(1):19–28, 2001.

[20] K. Higuchi, Y. Ishiguro, and J. Rekimoto. Flying eyes: free-space content creation using autonomous aerial vehicles. In *Proc. CHI'11 Extended Abstracts on Human Factors in Computing Systems*, pages 561–570, 2011.

[21] A. S. Huang, S. Tellex, A. Bachrach, T. Kollar, D. Roy, and N. Roy. Natural language command of an autonomous micro-air vehicle. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 2663–2669, 2010.

[22] B. Jones, K. Dillman, R. Tang, A. Tang, E. Sharlin, L. Oehlberg, C. Neustaedter, and S. Bateman. Elevating communication, collaboration, and shared experiences in mobile video through drones. In *Proc. ACM Conf. on Designing Interactive Systems*, pages 1123–1135, 2016.

[23] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics (TOG)*, 34(6):238, 2015.

[24] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2012.

[25] E. Marder-Eppstein. Project tango. In *Proc. ACM SIGGRAPH'16 Real-Time Live!*, pages 40:25–40:25, 2016.

[26] D. E. McGovern. Experience and results in teleoperation of land vehicles. In *Pictorial communication in virtual and real environments*, pages 182–195, 1991.

[27] F. F. Mueller and M. Muirhead. Jogging with a quadcopter. In *Proc. ACM Conf. on Human Factors in Computing Systems*, pages 2023–2032, 2015.

[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[29] G. Nebehay and R. Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2784–2791, 2015.

[30] H. Nozaki. Flying display: a movable display pairing projector and screen in the air. In *Proc. CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 909–914, 2014.

[31] M. Obaid, F. Kistler, G. Kasparavičiūtė, A. E. Yantaç, and M. Fjeld. How would you gesture navigate a drone?: a user-centered approach to control a drone. In *Proc. Int. Academic Mindtrek Conf.*, pages 113–121, 2016.

[32] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 2609–2616, 2014.

[33] F. Sadeghi and S. Levine. (CAD)^{2}RL: Real single-image flight without a single real image. In *Proc. Robotics: Science and Systems*, 2017.

[34] J. Scheible, A. Hoth, J. Saal, and H. Su. Displaydrone: a flying robot based interactive display. In *Proc. ACM Int. Symposium on Pervasive Displays*, pages 49–54. ACM, 2013.

[35] T. B. Sheridan. Teleoperation, telerobotics and telepresence: A progress report. *Control Engineering Practice*, 3(2):205–214, 1995.